

第4章 早わかり車載ネットワーク
――標準規格CAN，LIN，FlexRayの概要を押さえる

佐藤道夫

組み込みシステムでは，マイコンと周辺回路，あるいは制御機器と周辺機器をつなぐ通信プロトコルとしてさまざまな規格が存在する．自動車においても，搭載されているECU（電子制御ユニット）どうしを結ぶためにネットワークを利用する．従来は各自動車メーカ独自の規格が多かったが，最近では開発期間，コストなどを抑えるため，車載用の通信プロトコルの標準化が進んでいる．ここでは，制御系の車載ネットワーク・プロトコルとして，CAN，LIN，FlexRayについて，アクセス方式やフレーム構造などの基礎知識をまとめる．（編集部）

ここでは，現在の制御系ネットワークのデファクト・スタンダード（業界標準）とも言える「CAN（controller area network）」，ボディ系ネットワークへの採用が期待される「LIN（local interconnect network）」，X-by-wireアプリケーション注1に対応した「FlexRay」について，それぞれの特徴をまとめていきます（**表1**）．

注1：ステアリング制御やブレーキ制御，サスペンション制御など，従来機械的に制御していたものを，電子的制御に置き換えることで，自動車の軽量化（燃費の改善）やデザインの自由度の向上などを図る．

● 知っている知識をもとに新規格を理解する

ところで，新しいネットワーク・プロトコルを効率良く理解するには，どのようにすればよいのでしょうか．筆者の場合，LAN（local area network）で利用されているEthernetについての知識があるので，これをベースにメッセージの送信ルール（メディア・アクセス方式）などを確認します．

Ethernetのメディア・アクセス方式はCSMA/CD（carrier sense multiple access with collision detection）ですが，例えばCANはこの方式を採用しています．ここで問題になるのが，Ethernetは情報系のネットワーク・プロトコルであるという点です．なぜなら，CANは自動車

表1 CAN，LIN，FlexRayの概要

プロトコル	CAN 2.0	LIN	FlexRay
アプリケーション	パワートレイン系 制御，診断	ボディ系 スマート・コネクタ	X-by-wire 安全制御
伝送媒体	2線式（より対線）	1線式	2線式（より対線）， （光ファイバ注）
メディア・アクセス	イベント・ドリブン （マルチマスタ）	タイム・トリガ （マスタ-スレーブ）	タイム・トリガ
エラー検出	15ビットCRC	ビット・チェック・サム	24ビットCRC
ID（識別子）長	11ビット（CAN 2.0A） 29ビット（CAN 2.0B）	8ビット（1バイト）	11ビット
データ長	0～8バイト	8バイト	0～254バイト
メッセージ・アクノリッジ	あり	なし	なし
最大ビット・レート	10kbps～1Mbps	1kbps～20kbps	10Mbps
最大バス長	指定なし（平均40m）	40m	24m
最大ノード数	指定なし（平均32）	16	22
マイコンの必要性	要	要（マスタ）	要
スリープ/ウェイクアップ	なし	あり	あり
ハードウェア	あり	なし	あり

注：光ファイバについては，現在のプロトコルでは未定義

では制御系ネットワークに利用されるため,「リアルタイム性(ある処理に対する要求が出されてから,実際にその処理が行われるまでの時間)」が重要なポイントとなります.このリアルタイム性が,はたして情報系ネットワークと同じプロトコルで確保できるのかが問題となるわけです.

Ethernetでは,同時に複数のノード(ネットワーク上のコンピュータ)がメディア(通信媒体)にメッセージを送信することを想定しています.このような状態を「衝突(collision)」とし,これを「検出(detection)」することを定めています.衝突を検出した場合は,ある時間待機した後,メッセージを再送します.しかし,ある時間待って再送しても,また衝突が起こる可能性は少なからずあるため,長い時間通信できないノードが出てくるということが懸念されます.では,CANはCSMA/CDというプロトコルを自動車の制御ネットワークに利用するために,どのようにこの問題を解決したのでしょうか.

自動車向けのネットワーク特有のリアルタイム性や信頼性,エラー処理などを踏まえながら,車載ネットワーク・プロトコルの概要を見ていきましょう.

CANプロトコルの概要

車載ネットワーク・プロトコルで最初のデファクト・スタンダードになったのは,ドイツのBosch社が開発した「CAN」です.同社は,1980年代の初期に「将来,自動車にもネットワークが使われる」という判断から研究開発をスタートし,1991年に現在の仕様であるCANプロトコル仕様2.0A/Bを公開しました.1993年にはISOの「ISO11989」として標準化されました.

● ネットワーク・トポロジ――バス型

CANのネットワーク・トポロジは,基本的にバス型です.

Bosch社の策定したCAN 2.0A/B仕様書にはネットワークのバス長は規定されていません.また,接続可能なノード数についても規定されていませんが,これらはCANの物理層の処理を行うトランシーバ(送受信回路)の電気的仕様で決まってきます.例えば,**図1**に示す高速CAN(J2284)は米国の自動車関連の業界団体であるSAE(Society of Automotive Engineers)が策定したCANの規格ですが,ここではビット・レートは500kbps,最大バス長は40m,最大ノード数は16と規定されています.

● フレーム――データ,リモート,エラー,オーバロード

CANのメッセージ転送では,4種類のフレームが用意されています.

- **データ・フレーム(図2(a))**:トランスミッタ(送信ノード)からレシーバ(受信ノード)にデータを伝えるフレーム.データ・フレームと次のデータ・フレームの間は,3ビット以上のインターフレーム・スペースというフレームで分離される.
- **リモート・フレーム(図2(b))**:バス上のあるノードからデータ・フレーム送信をリクエストするときに送信されるフレーム.このとき,データ・フレームのIDとリモート・フレームのIDは同じ.データ・フレームとリモート・フレームの間は,インターフレーム・スペースで分離される.
- **エラー・フレーム(図2(c))**:バスのエラーを検出したノードによって送信されるフレーム.
- **オーバロード・フレーム(図2(d))**:先行するデータ・フレームとそれに続くデータ・フレームまたはリモート・フレームの間に余分な遅延を供給するために使用されるフレーム.レシーバが次のデータ・フレームまたはリモート・フレームの遅延を要求し,インターフレーム・スペースの中にドミナントを検出した場合に送信される.

ここで,CANのバス・レベルについて少しお話ししておきましょう.CANのバス信号には,「ドミナント」と「リセッシブ」の二つのレベルがあります.ドミナントはリセッシブより優先順位が高く,バスにつながるノードが一つでもドミナントだと,バスはドミナント・レベルになります.

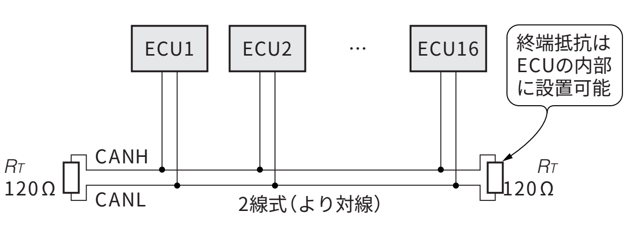


図1 CANのトポロジ例
基本的にバス型のトポロジとなる.CANの物理層については数種類の異なる規格が策定されているので,接続可能ノード数や伝送速度などは物理層の処理を行うトランシーバの電気的仕様で決まる.図は高速CAN(J2284)の例.

4

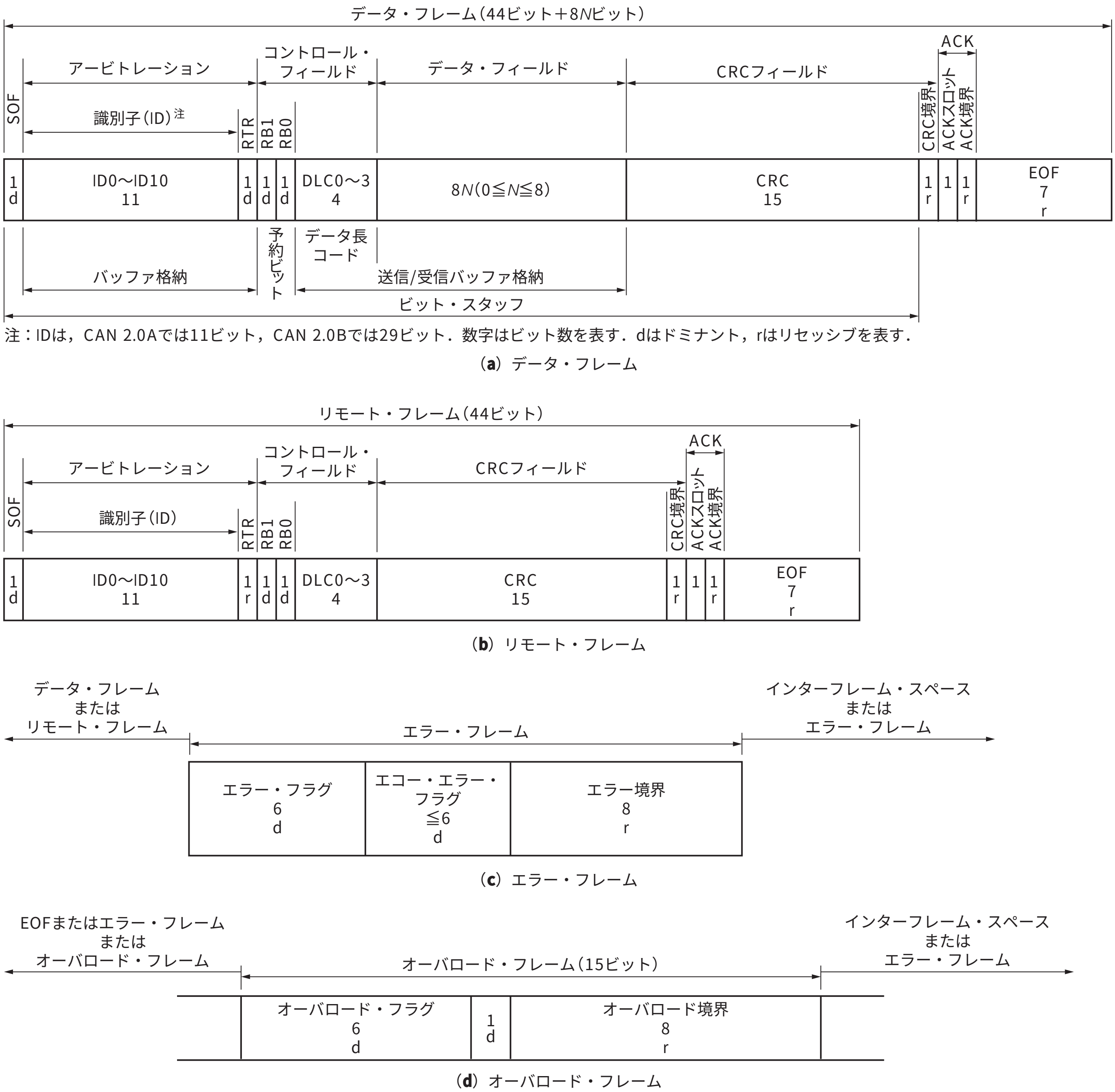


図2　CANのメッセージ転送に用いられるフレーム

CANのメッセージ転送では，4種類のフレームが用意されている．同じIDを持つリモート・フレームとデータ・フレームが同時に送信された場合，RTR（Remote Transmission Request）ビットでアービトレーションをとる．データ・フレームのRTRビットはドミナント，リモート・フレームのRTRビットはリセッシブなので，データ・フレームが優先される．（c）のエラー・フレームは，エラーを検出したすべてのノードが送信するフレームだが，バスには遅延があるため，送信ノードから見て近端のノードと遠端のノードの間に送信のずれが発生する．エコー・エラー・フラグは，そのずれを最大6ビットまで許容するためのもの．

データ・フレームとリモート・フレームにはACK（アクノリッジ）ビットが用意されています．ACKビットは，それぞれのフレームが正しく受信されたときに，受信ノードから返ってきます．データ・フレーム，リモート・フレームともに送信時のACKビットのレベルは「リセッシブ」です．受信ノードが正常に受信すると，このビットを「ドミナント」に変換して応答します．送信ノードはACKビットが「ドミナント」になったので受信ノードは正常に受信したと判断します．

● メディア・アクセス――イベントで送信を開始

CANのアクセス方式はイベント・ドリブン（事象駆動）

とも呼ばれ，なんらかのイベントが生じることでメッセージを送信します．

また，CANはEthernetと同じCSMA/CDをベースとしており，バスが開放（信号がない状態．「アイドル状態」とも言う）されているときは，どのノードでもメッセージの送信を開始できるマルチマスタ方式です．先に，このマルチマスタ方式のため，Ethernetではリアルタイム性について問題があると述べました．CANでは，「アービトレーション（調停）」というプロセスを入れることで，この問題を解決しています．このアービトレーションについて，以下に例を挙げて説明します．

図3は，ネットワークに三つのノードが接続されており，同時にメッセージを送信している状態を示しています．**図2**に示したように，CANのメッセージ・フレーム（データ・フレーム，リモート・フレーム）は"SOF（start of frame）"で始まります．このとき，バス・レベルはドミナントです．

次は識別子（ID）ですが，このIDフィールドのMSB（most significant bit）からアービトレーションが始まります．各ノードのMSB（**図3**では10ビット目）を比較すると，いずれのノードもリセッシブ・レベルで，バスもリセッシブ・レベルとなっています．各ノードはバス・レベルをモニタしています．**図3**において，6ビット目まではそれぞれのノードのメッセージ・レベルとバス・レベルが同じなので，自分自身が送信していると判断しています．次に，5ビット目を見ると，ノード2はリセッシブ，ノード1とノード3はドミナントとなっています．ここで，アービトレーションが行われます．

IDフィールドでは，送信ビットごとに論理積（入力に一つでも'0'があれば，出力は'0'）をとってアービトレーションを行います．例えば，**図3**の場合は以下の式となります．

バス・レベル＝（ノード1のID_n）and（ノード2のID_n）and（ノード3のID_n）

ここで，ドミナントの論理値は'0'です．つまり，各ノードの5ビット目の論理積をとると，バス・レベルは'0'，つまりドミナントになります．ノード2はバス・レベルと自分のメッセージのレベルが違うので，すぐに送信を止めます．同じ要領で，2ビット目ではノード1とノード3がアービトレーションされ，最終的にノード3がメッセージを最後まで送信できることになります．

アービトレーションによってメッセージの送信を止めた

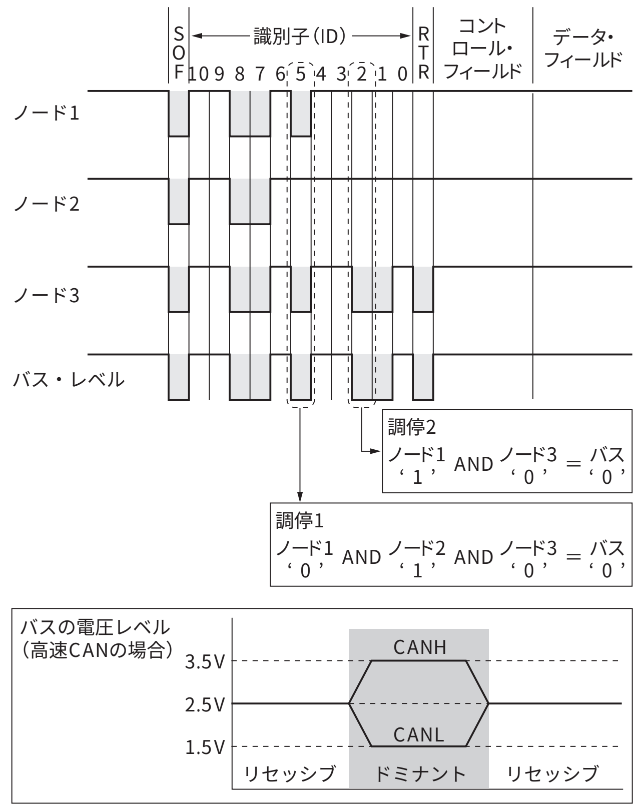


図3　CANのアービトレーション

IDフィールドでアービトレーションをとる．各ノードの送信ビットごとに論理積をとって，モニタしているバス・レベルと同じであれば送信し，異なれば送信を停止して，送信可能な状態になるまで待機する．

ノード1とノード2は，ノード3が送信を終了すると直ちに再送を開始します．再度ノード3が送信を開始しないかぎり，最初にノード1が送信し，その後ノード2が送信してアービトレーションが終了します．

このように，CANはEthernetと同じCSMA/CD方式を採用しているのですが，リアルタイム性に優れています．

● エラー検出と処理――不ぐあいノードを封じ込める

CANでは，次のようなエラーを検出します（**図4**）．

- **ビット・エラー**：送信ノードが送信しているビット値と違う値を検出したとき
- **スタッフ・エラー**：同じ値のビットを連続して6ビット以上受信したとき
- **CRCエラー**：受信ノードがCRC計算して，計算値がまちがっていたとき
- **フォーム・エラー**：受信ノードが規定フォームと異なるビットを検出したとき
- **アクノリッジ・エラー**：送信ノードがアクノリッジのド

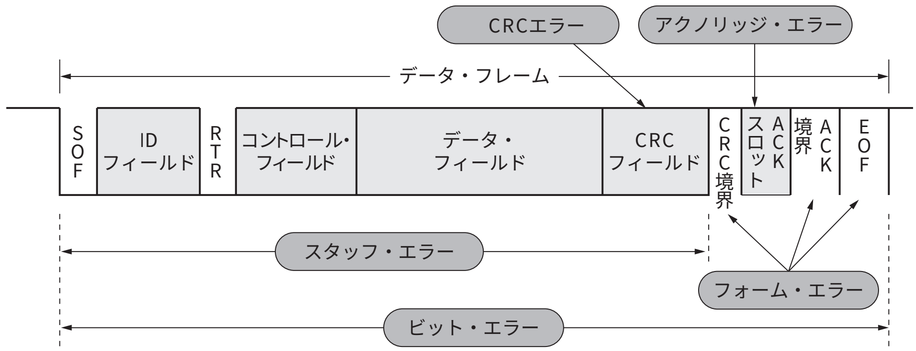


図4
CANのエラー検出
CANでは，ビット・エラー，スタッフ・エラー，CRCエラー，フォーム・エラー，アクノリッジ・エラーという五つのエラーを検出する．

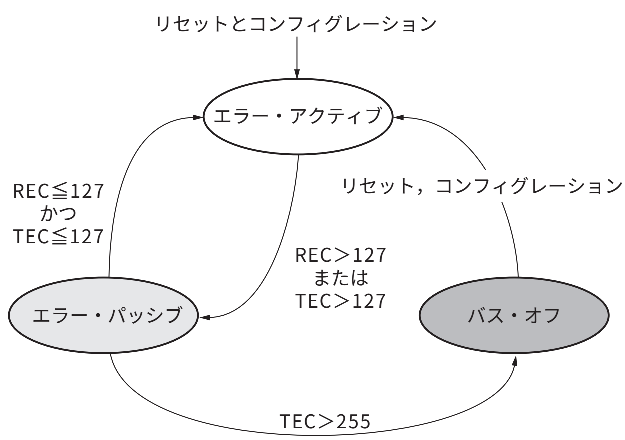


図5　不ぐあいノードの封じ込め
バス上に接続されたノードは，トランスミット・エラー・カウンタ（TEC）とレシーバ・エラー・カウンタ（REC）を内蔵している． それぞれのカウンタ値が127を超えると，通常のフレーム送受信状態（エラー・アクティブ）からエラー・パッシブという状態に遷移するが，送受信は継続している．エラー・パッシブに入った後，エラーがなければカウンタ値が下がりエラー・アクティブに戻るが，エラーが発生し続けるとカウンタ値が上がり，バス・オフ（フレーム送信停止）状態になる．

ミナント・ビットを検出できないとき

各ノードは，こうしたエラーを検出したとき，エラー・フラグを送信します．

CANの各ノードはREC（Receive Error Counter）とTEC（Transmit Error Counter）という二つのカウンタを内蔵しています．エラーを検出すると，その内容によってカウンタの値を増減します．**図5**に示すように，RECとTECのカウンタ値によって三つの状態，すなわち，「エラー・アクティブ」，「エラー・パッシブ」，「バス・オフ」に遷移します．例えば，不ぐあいノードがネットワーク上に異常な信号を出し続けたとき（「bubble idiot」と呼ばれる），各カウンタが255に達するとそのノードはネットワークから切り離されるバス・オフ状態になります．このようにして，エラーのあるノードがネットワークに対して悪い影響を与えないようにします．

● 同期――ビットごとに位相エラーを補正する

CANネットワークでは各ノードは非同期で動いています．そのため，ノードどうしの同期の実現方法，およびそれによって各ビットが正しく判定できるようなしくみが必要になります．

CANのネットワークの伝送速度は5kbps～1Mbpsの可変レートです．例えば，この伝送速度を500kbpsに設定すると，1ビット当たりの伝送時間（1ビット・タイム）は2μsです．この2μsは，次の四つのセグメントで構成されています（**図6**）．

- **SYNC_SEG**：バス上の各ノードの同期をとる
- **PROP_SEG**：ネットワークの物理的な遅延時間を補う
- **PHASE_SEG1，PHASE_SEG2**：エッジ位相エラーを補う

各ビットの立ち下がりエッジがSYNC_SEGのセグメント内に入っているかどうかで同期を判定します．このセグメントに入っていれば同期がとられていることになります．ビットの立ち下がりがSYNC_SEGより前であればPHASE_SEG2が縮小し，SYNC_SEGより後であればPHASE_SEG1が拡張して再同期を行います．このプロセスは，立ち下がりが発生するたびにビット単位で行われます．

LINプロトコルの概要

CANはおもにパワートレイン系やシャーシ系のネットワークの標準規格として採用されています．一方，比較的低

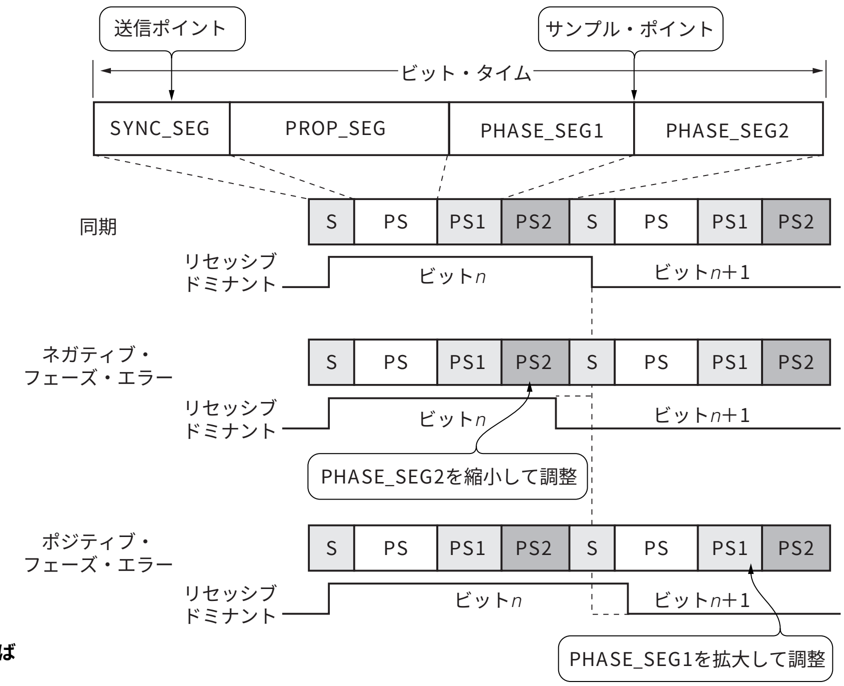


図6
CANのビット・タイミング
SYNC_SEGで各ビットの立ち下がりが起こっていれば同期している．

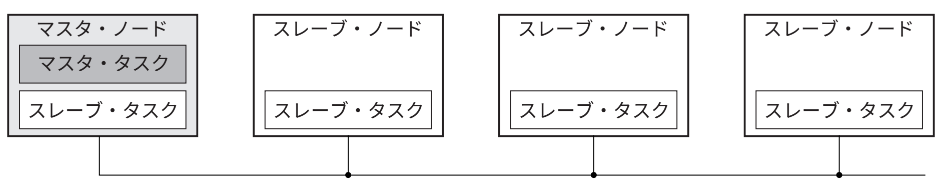


図7
LINトポロジ
基本的にバス型トポロジをとる．1個のマスタ・ノードに最大15個のスレーブ・ノードが接続される．

速なボディ系では，以前はCANのようなデファクト・スタンダードと呼べるプロトコルがなく，各自動車メーカはUARTなどをベースとした独自のネットワーク・プロトコルを開発して搭載していました．このような中，欧州の自動車メーカは，将来，車載ネットワークにさまざまなプロトコルが存在することになり，その開発や保守などが問題になると考えていました．そこで，低速で低コストのネットワークを標準化するため，2000年3月に正式に「LINコンソーシアム」を立ち上げました．

LINの規格では，通信プロトコル，メディア（通信媒体），開発ツール間のインターフェース，API（application programming interface）などが定義されています．また，LINは低コストで対応機器を製造できることを目標としており，以下の点を基本としてプロトコルが構成されます．

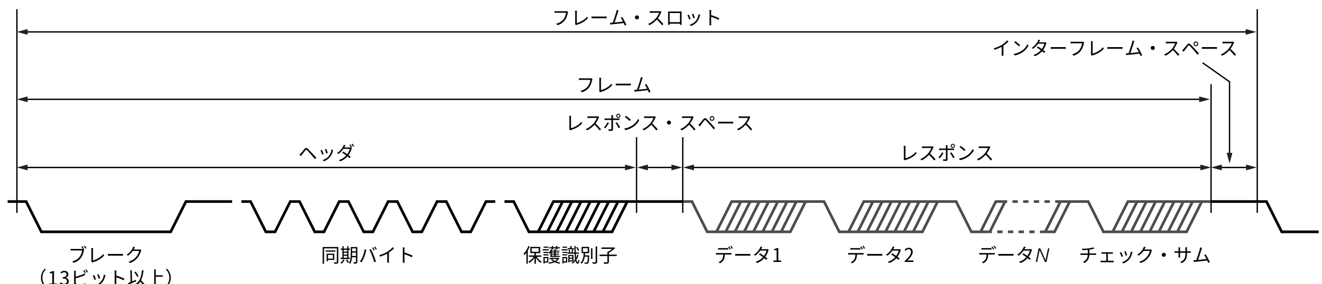
- 通信方式として，すでに普及しているUART/SCIインターフェースを利用する
- アクセス方式は，単一マスタと複数スレーブで構成される．マスタがネットワークのスケジュール管理を行う
- スレーブ・ノードは，マスタからの通信データの中に基準クロックをのせており，水晶発振子やセラミック共振子を使用しなくても同期がとれる
- 単一ワイヤで動作し，通信速度は最大20kbps

現在，LINの最新バージョンはLIN 2.0です．LIN 2.0は，従来のバージョンであるLIN 1.3に対して，信頼性を向上させたり，LIN対応機器のプラグ・アンド・プレイを可能にするような機能が追加されています．ここでは，LINの基本的な概要を理解していただくため，LIN 1.3プロトコルにそって，話を進めます．

● ネットワーク・トポロジ――バス型

LINプロトコルでは，とくにネットワーク・トポロジについての規定はありませんが，基本的にバス型トポロジをとります（**図7**）．LINネットワーク（クラスタ）は，マスタ・タスク[注2]を実行するノード（マスタ・ノード）が一つ

注2：マスタ・タスクは，いつ，どのようにフレームをバスに転送させるかを決める．



(a) LINフレーム

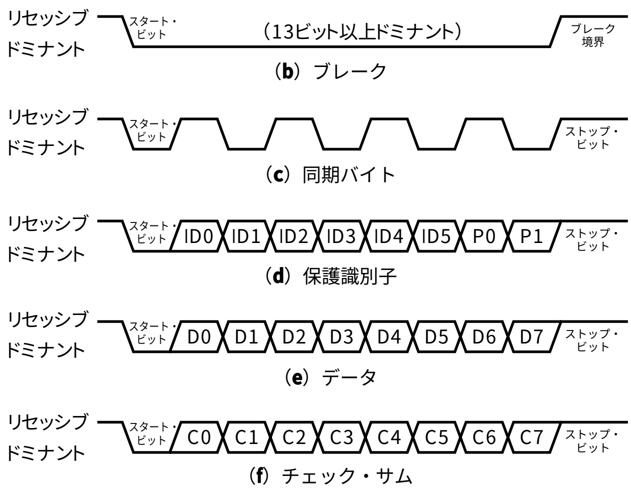


図8 LINフレームの構造
フィールドと各フィールドにおけるフレーム構造を示す．保護識別子のビット0～5は識別子として予約されており，0～63の範囲で利用できる．ビット6～7はパリティ．

と，スレーブ・タスク注3だけを実行するノード(スレーブ・ノード)が最大15個接続されます．つまり，クラスタとして最大16ノードを超えてはならないと規定されています．また，マスタ・ノードの指示がないかぎり，スレーブ・ノードは機能しません．

LINノードはトランシーバを介して物理的なバス配線に接続されます．

● フレーム――マスタのヘッダとスレーブのレスポンス

LINのフレームは，マスタ・タスクから出力される「ヘッダ」と，スレーブ・タスクから出力される「レスポンス」で構成されます．ヘッダは，**図8**に示すように13ビット以上のドミナント・レベル(つまり‘0’)で構成されるブレーク(Sync Breakフィールド)，0x55(16進数)のデータ値を有する同期バイト(Syncフィールド)，6ビットのIDと2ビットのパリティを持つ識別子(IDフィールド)からなります．

このヘッダに対して，スレーブ・タスクが最大8バイトのデータ(Dataフィールド)と1バイトのチェック・サム注4(Check Sumフィールド)で構成されるレスポンスを返します．

基本的に，ブレークと同期バイト以外はUART/SCIのフレーム・フォーマットを使用しています．また，CANのように送信内容ごとのフレームが用意されていないので，IDに意味を持たせて代替しています．

● メディア・アクセス――タイム・トリガによる通信実行

LINのアクセスは，マスタ・ノードがヘッダを送信することで開始します．

イベント・ドリブンのCANに対して，LINはマスタ・タスクによってメッセージの送信を開始します．LINの送信の開始のきっかけとなるのはイベントではないので，マスタ・タスクがどのように実行するのかについて，あらかじめスケジュールを組んでおかないと通信が成立しません(**図9**)．このスケジュールは，自動車が動いている間中，繰り返し実行されます．このような方式を「タイム・トリガ」と呼びます．

LINの場合，CANのようにアービトレーションの機能を備えていません．それは，各タスクは時間で管理されているので，同期がとれているかぎりメッセージの衝突は発生しないからです．タスクの優先順位はマスタのスケジューラで決まります．**図10**に，基本的なLINの通信を示します．**図10(a)**はマスタ・ノードからスレーブ・ノード，およびスレーブ・ノードからマスタ・ノードへのデータの送信を，**図10(b)**はマスタ・ノードから複数スレーブへの

注3：スレーブ・タスクは，それぞれのフレームごとに転送するデータを準備する．
注4：データ伝送におけるエラー検出の一つ．あらかじめ送信するデータの合計値を計算しておき，この値をデータとともに送信する．受信側では受け取ったデータの合計値を計算し，送られてきた合計値と比較する．合致していなければ，エラーありということになる．なお，LIN 1.3ではデータについてのみチェック・サムをとっていたが，LIN 2.0はIDフィールドについてもチェック・サムをとる．

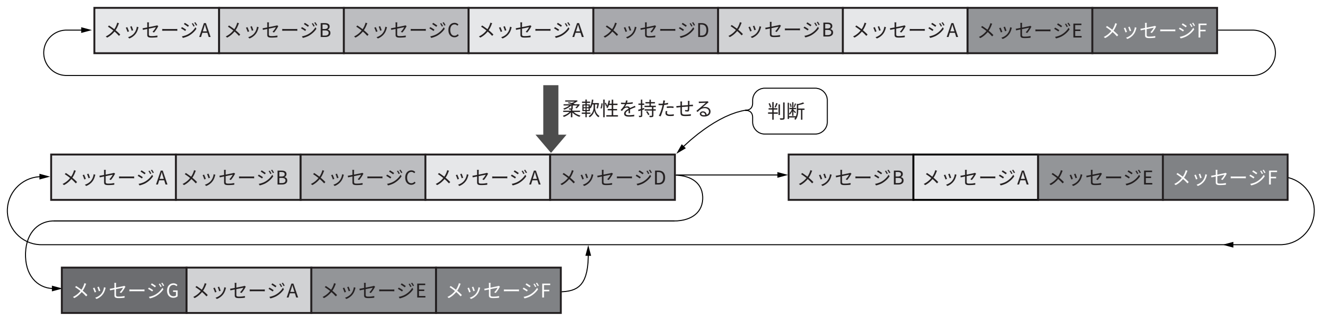


図9　LINのタイム・スケジュール

LINのアクセス方式はタイム・トリガ．マスタがスケジュール・テーブルで管理しているので，メッセージ・シーケンスはあらかじめ明確となる．マスタ側は，LINクラスタとしてアプリケーションを実行したり，ネットワークを管理する機能を持っている．また，アプリケーションによっては，スケジュールを変更して別の処理を行わなければならないケースも考えられる．そういうときのためにも，さまざまなことを想定したスケジュール表を用意しておき，そのための判断するルーチンを準備することが重要．

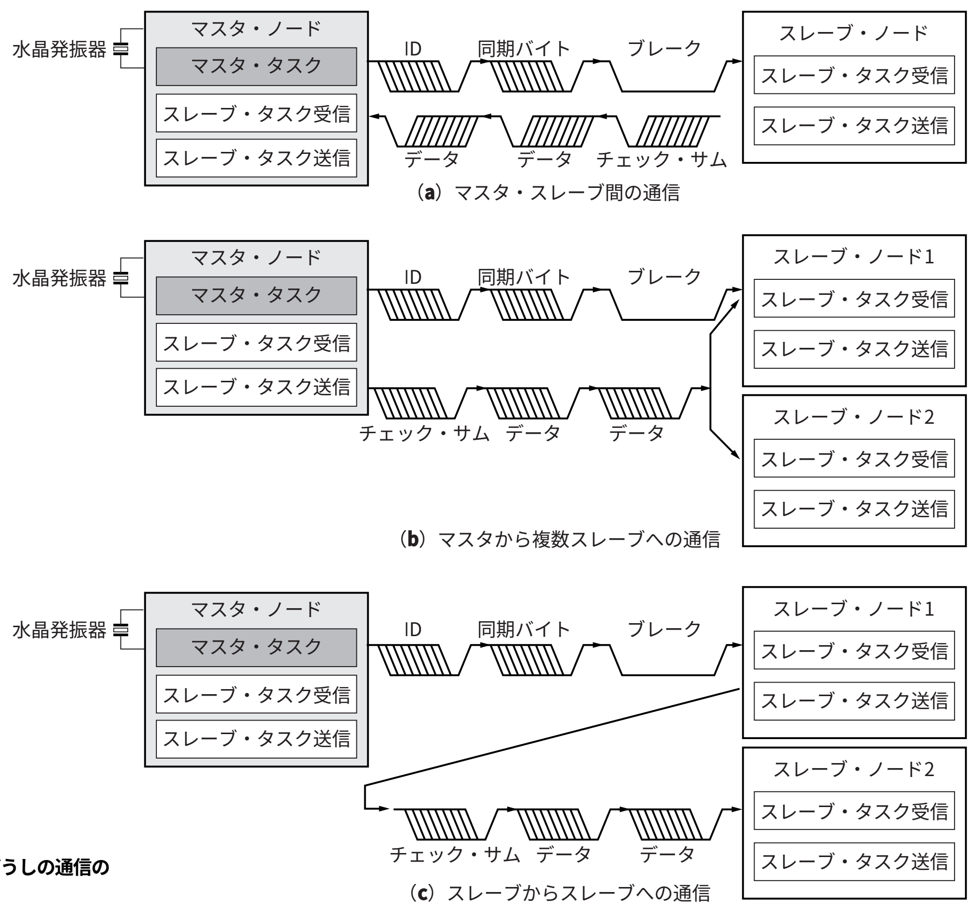


図10
LIN通信の例

基本的なマスタ・スレーブ間，スレーブどうしの通信の例を示す．

データ送信を，**図10(c)**はスレーブ・ノードからスレーブ・ノードへのデータの送信を示します．

LINにはCANのようなACKスロットが割り当てられていません．必要な場合はアプリケーション・ソフトウェアで実行することになります．また，データ長を表すビットもありません．このように，CANと比べるとシンプルな構造になっていることがわかります．

● エラー検出と処理――最新版では三つの状態を見るだけ

LINの場合，従来のLIN 1.3と最新バージョンのLIN 2.0ではエラー検出処理には大きな違いがあります．LIN 1.3では，

- ビット・エラー
- チェック・サム・エラー
- IDパリティ・エラー

- スレーブ・レスポンスなしエラー
- 同期フィールド不一致エラー
- 物理的バス・エラー

などが定義されていますが，LIN 2.0では定義されていません．

LIN 2.0では，レスポンスの中にステータス・ビット“Response_Error”を入れてマスタ・ノードに送信しています．このステータス・ビットによって，マスタ・ノードが次の三つの状態を判断します．

- Response_Error＝偽：ノードは正しく動作中
- Response_Error＝真：ノードが間欠性の問題を持つ
- ノードが返答しない：ノードが深刻な問題を持つ

不ぐあいノードの処理については，LIN 1.3，LIN 2.0ともにプロトコルでは定義されていません．そのため，システム・レベルで考えておく必要があります．

ビットの判定はUART/SCIフォーマットに沿っています．1ビット・タイムを16分割して，その分割した中心でサンプリングします．このとき“L”レベルであれば「0」，ハイレベルであれば「1」と判定します．

● 同期――タイム・トリガなのでクロック精度を規定

LINでは，マスタ・ノードのクロック発振器の誤差を最大±0.5%に，スレーブのクロック誤差を最大±2.0%に規定しています．その上で同期フィールドのパターン（0x55）を利用して同期をかけるようになっています．

FlexRayプロトコルの概要

高速CANがデファクト・スタンダードになって十数年がたちます．この間，自動車に搭載される機能の増加に伴い，ネットワーク・トポロジの複雑化やアプリケーションの相互接続性の問題，通信データ量の増大などの問題が表面化してきました．このような課題を解決するために，FlexRayが策定されています．最新バージョンであるFlexRay 2.0プロトコルは2004年7月に公開されたばかりです．

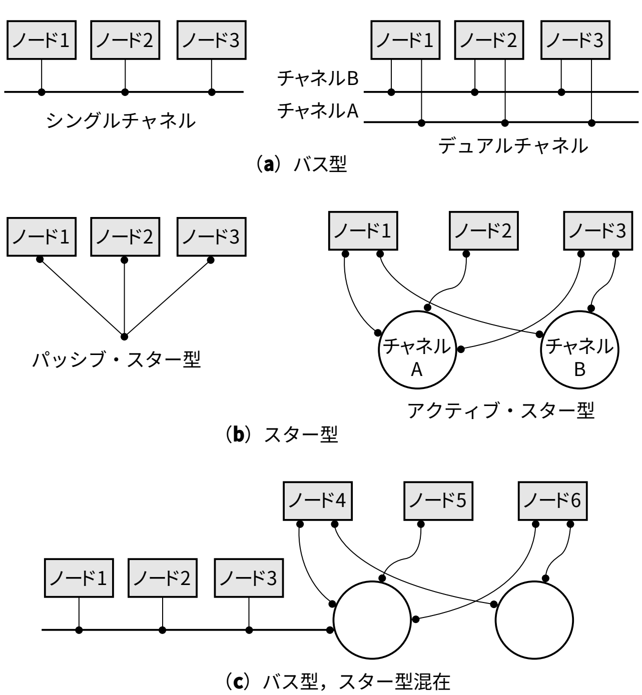


図11　FlexRayネットワーク・トポロジ
ローエンドからハイエンドの車種に対応できるように，バス型，スター型，バス・スター型混在のトポロジをとることが可能．

● ネットワーク・トポロジ――バス，スターの両方に対応

FlexRayのネットワーク・トポロジは，バス型，スター型のほか，バス型とスター型を混在させたものも利用可能です（**図11**）．バス型はCANやLINでも採用されているので設計面での経験値が高く，コスト効率も良いのですが，あまり高速なデータ転送は行えません．そこで，高速データ転送やエラー制御を向上させたい場合，スター型トポロジを採用することになります．

また，FlexRayではシングルチャネル（単一チャネル）とデュアルチャネル（二重チャネル）に対応しています．ワイヤ・ハーネスの量も少なく，低コスト化を図る場合はシングルチャネルで，フォールト・トレランス（耐故障性）を上げたい場合はデュアルチャネルでネットワークを構築します．

このように，車種に応じてトポロジやノードの接続などを変えることができるため，開発コストや部品調達コストを抑えられます．

● フレーム――フォーマットは一つだけ

FlexRayのフレーム構造は，**図12**に示すように1種類だけです．CANのように何種類も定義されていません．

FlexRayのフレームは，ヘッダ・セグメント，ペイロード・セグメント，トレーラ・セグメントの三つのセグメン

トで構成されています．ペイロード・セグメントは，0〜254バイトのデータのほか，オプションで16ビットのメッセージIDを挿入する場合もあります．

また，アクノリッジについては，定義されていません．

● メディア・アクセス――フレーム送信を時間で規定

FlexRayのメディア・アクセス方式はLINと同じタイム・トリガです．つまり，あらかじめメッセージ送信についてスケジューリングされています．

FlexRayのメディア・アクセスにおいて，もっとも大きなアクセス単位はコミュニケーション・サイクルです（**図13**）．このコミュニケーション・サイクルによって一つのノードの送信にかかる時間が決まります．コミュニケーション・サイクルは，STDMA（static time division multiple access）と呼ばれるアクセス方式を用いる静的セグメント，FTDMA（flexible time division multiple access）と呼ばれるアクセス方式を用いる動的セグメント，同期やエラー訂正などで用いられるネットワーク・アイドル時間（NIT）で構成されます注5．

静的セグメントの各スロット（静的スロット）は，一度長さ（時間）を決めたら，送信フレームの有無に関係なく一定です．一方，動的セグメントでは，フレームが送信される

注5：オプションとして「シンボル・ウィンドウ」が用意されている．スタートアップ時，ウェイク・アップ時などで使われる．

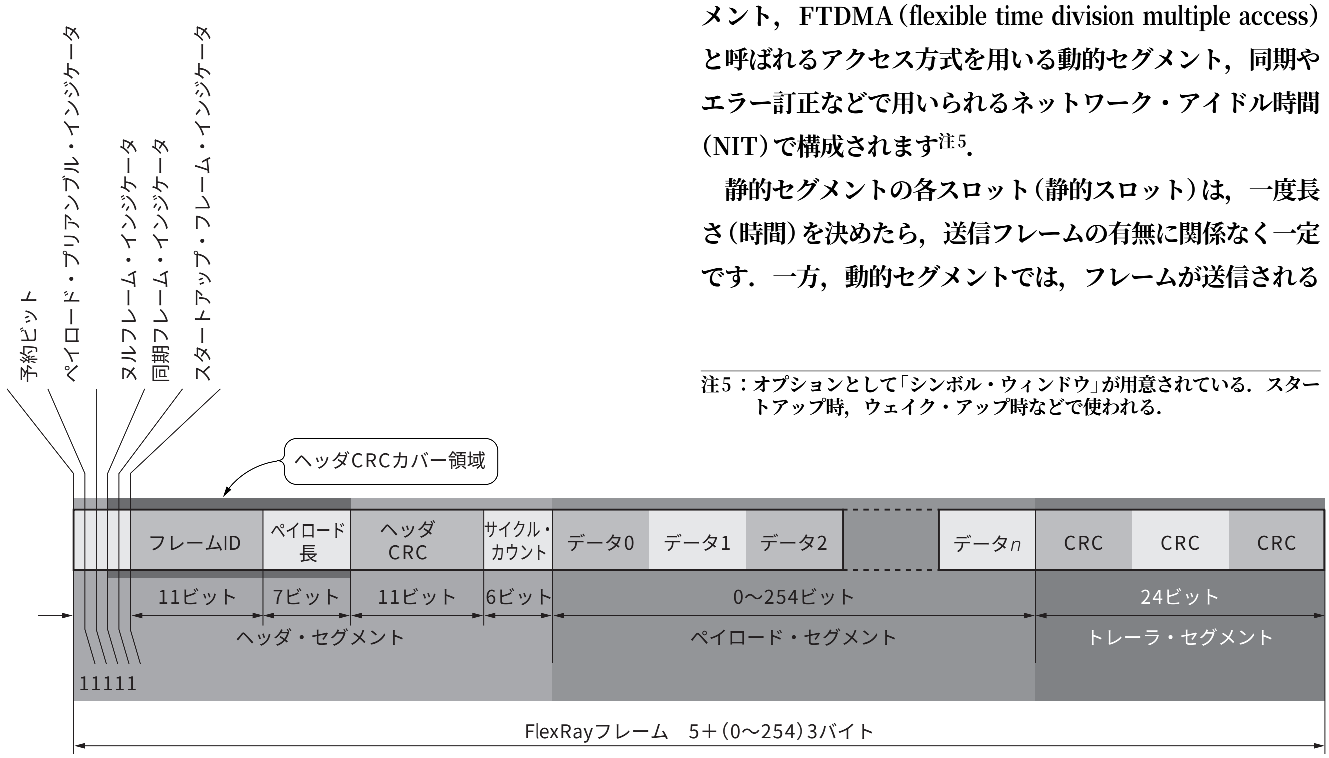


図12 FlexRayのフレーム構造

FlexRayのフレームは，ヘッダ・セグメント，ペイロード・セグメント，トレーラ・セグメントの三つのセグメントで構成されている．トレーラ・セグメントには，ヘッダ・セグメントとペイロード・セグメントをCRC計算したチェック・コードが入っており，フレーム全体のエラー・チェックの機能を果たす．

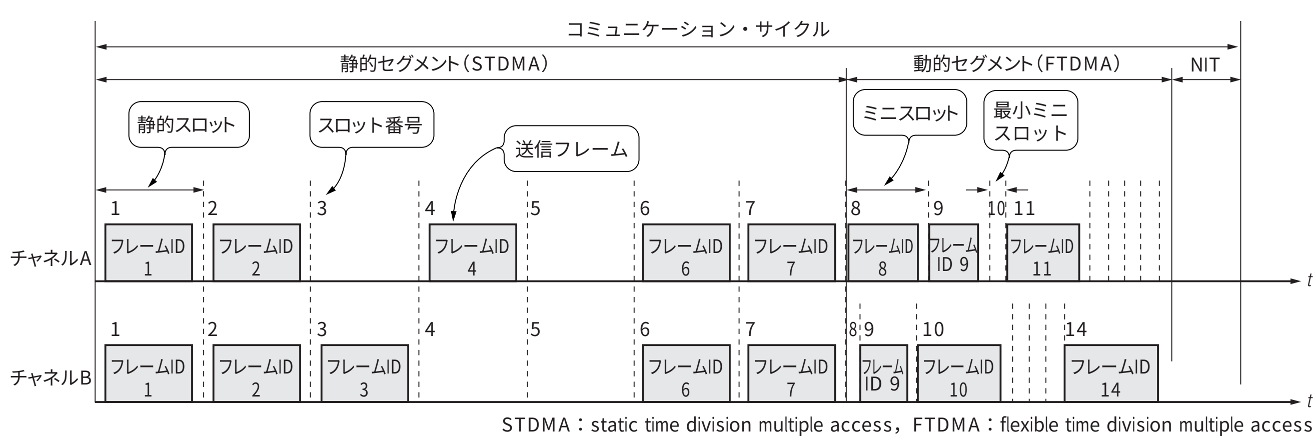


図13 FlexRayのメディア・アクセス制御

静的セグメント，動的セグメント，ネットワーク・アイドル時間（NIT）は，あらかじめ決められた長さに固定される．

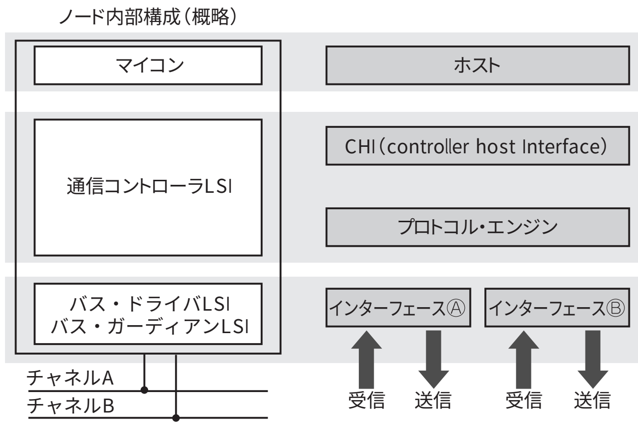


図14　FlexRayアーキテクチャの概要
FlexRayのアーキテクチャはホスト，CHI，プロトコル・エンジンなどで構成される．このアーキテクチャは，マイコンや通信コントローラLSI，バス・ドライバLSI，バス・ガーディアンLSI（エラーの封じ込めを行う）などで実現される．

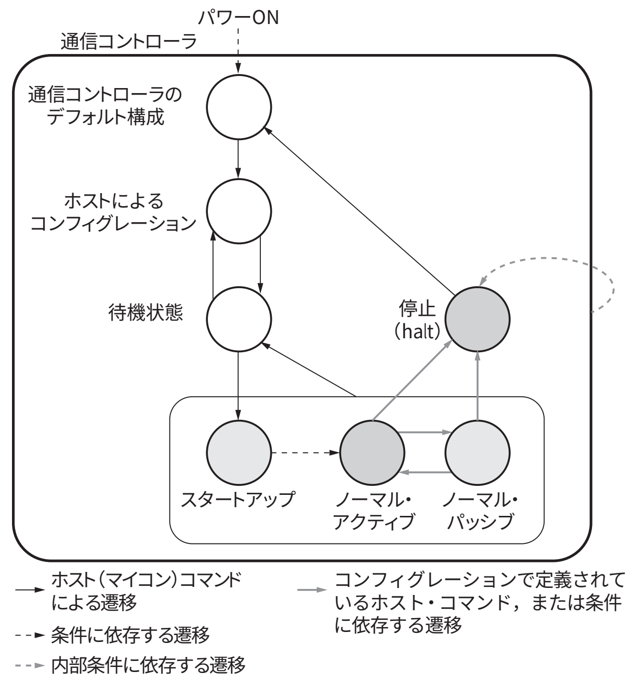


図15　通信コントローラの内部状態遷移の概要
FlexRayの仕様で定められている通信コントローラの内部状態遷移を一部抜粋したものを示す．

ときはスロット（ミニスロット）の長さが必要に応じて拡張されますが，送信フレームがない場合は最小のスロット幅となります．しかし，スロットの幅が変化しても動的セグメントの長さは固定です．また，ネットワーク・アイドル時間も固定です．このようにして，あらかじめフレームのスケジューリングを決定しておきます．

図13は，静的セグメントと動的セグメントの例も示しています．静的セグメントについては，チャネルAとチャネルBのデュアル送信時のすべてのパターン（チャネルAとB両方へのフレーム送信，片方へのフレーム送信，および送信なし）を表しています．ここで，フレームのID番号とスロット番号は一致します．動的セグメントは静的セグメントと異なり，フレームごと，およびフレームの有無によってスロットの幅が変化します．動的セグメントにおいても，フレームのID番号とスロット番号は一致します．このようにフレームIDとスロット番号によって同期がとれているかぎり，衝突が発生することはありません．

● エラー検出と処理――基本は「通信継続」

図14にFlexRayのアーキテクチャを，**図15**にFlexRayの仕様で定められている通信コントローラLSIの内部状態遷移の概要を示します．

図15において，通信コントローラLSIがパワーONすると，まず通信コントローラ自身のコンフィグレーション（初期設定）を行います．その後，ホスト（マイコン）によるノードとしてのコンフィグレーションを行い，これが終了すると待機状態に遷移します．スタートアップは通信の準備を行う状態ですが，準備が終了するとノーマル・アクティブ状態となって通常の通信を実行します．なんらかのエラーが発生したり，同期がとれなくなったときなどはノーマル・パッシブ状態に遷移することがあります．ノーマル・パッシブ状態に入ったとき，そのノードが送信中であればフレームの送信を中止します．受信中であれば，フレームの受信を継続します．クロック同期も継続しており，問題がなければノーマル・アクティブに戻ります．

なお，FlexRayでは，各ノードがエラー・カウンタを備えています．このカウンタ値によって（ただし，カウンタのしきい値は仕様で定義されていない）ノーマル・アクティブとノーマル・パッシブの状態を行き来します．

致命的なエラーを検出した場合は，停止状態（halt）に遷移しますが，FlexRayプロトコルではエラーに関する処理を詳細に規定しておらず，基本的には多少エラーが発生しても通信を継続するようになっています．通信を中止する決定は，プロトコルでではなくホストのアプリケーションで判断するように考えられています．

CANでは，不ぐあいノードがネットワーク上に異常な信号を出し続けた場合の不ぐあいノードの封じ込めをプロ

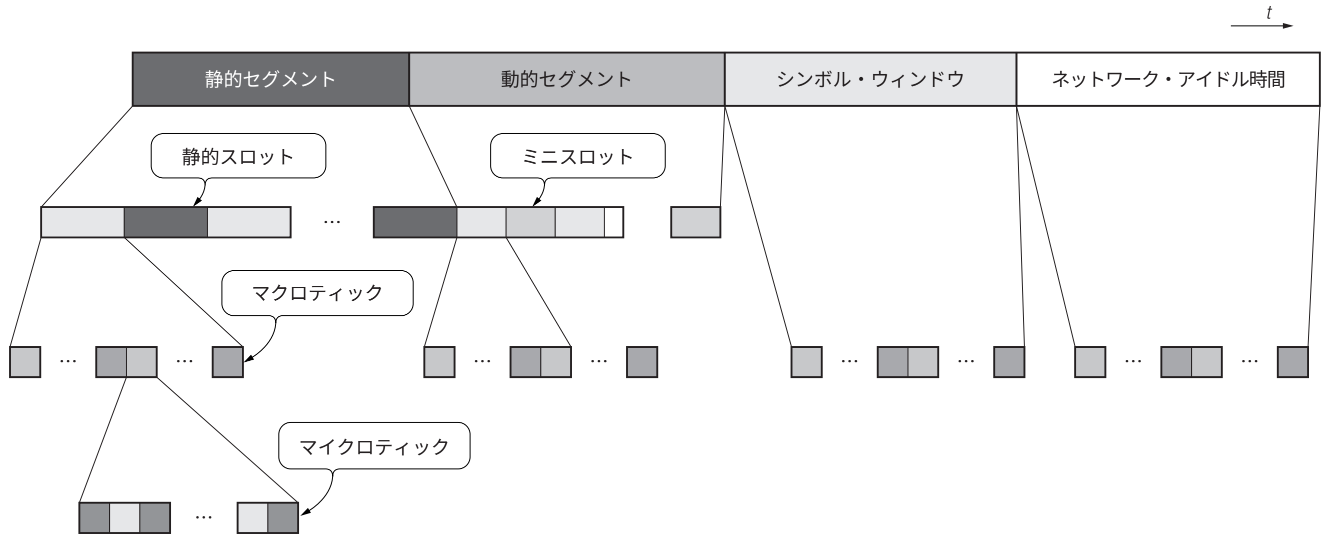


図16　マクロティックとマイクロティック

メディア・アクセスを階層的に示した図．マクロティックは，同期サービスを提供する機能を備えている．このマクロティックをさらに細かく分割したものがマイクロティックとなる．

トコルで処理しますが，FlexRayでは専用LSI（「バス・ガーディアン」と呼ばれる）によって実現します．

● 同期――コミュニケーション・サイクルごとに同期訂正

クロック同期はFlexRayにとってとても重要です．クロック同期には，オフセット訂正（位相訂正）とレート訂正（周波数訂正）の2種類があります．

各ノードの同期は，マクロティックをもとに実行されます．マクロティックとは，スロット（静的スロット，ミニスロット）やネットワーク・アイドル時間などを構成する時間単位です（**図16**）．コミュニケーション・サイクルごとに各ノード間のオフセットや経年変化による遅延誤差をこのマクロティックを用いて測定します．そして，次のコミュニケーション・サイクルで，マクロティックよりさらに細かい時間単位のマイクロティックで訂正します．

こうして同期が持続できるようにしています．

参考・引用*文献

(1) Christopher A. Lupini；Multiplex Bus Progression 2003，SAE 2003-01-0111.

さとう・みちお
フリースケール セミコンダクター ジャパン（株）

4

<筆者プロフィール>
佐藤道夫．1982年にモトローラに入社し，半導体事業部に所属．品質保証や営業を経験し，1992年からコントロール＆ネットワークのアプリケーション・エンジニアとして，ロンワークスの普及に従事する．現在は，フリースケール・セミコンダクタ・ジャパンの半導体セクター トランスポーテーション＆スタンダードプロダクツグループにおいて，主に車載向けのネットワークの技術担当マネージャをしている．